Avoiding Communication in Dense Linear Algebra

Grey Ballard

UC Berkeley

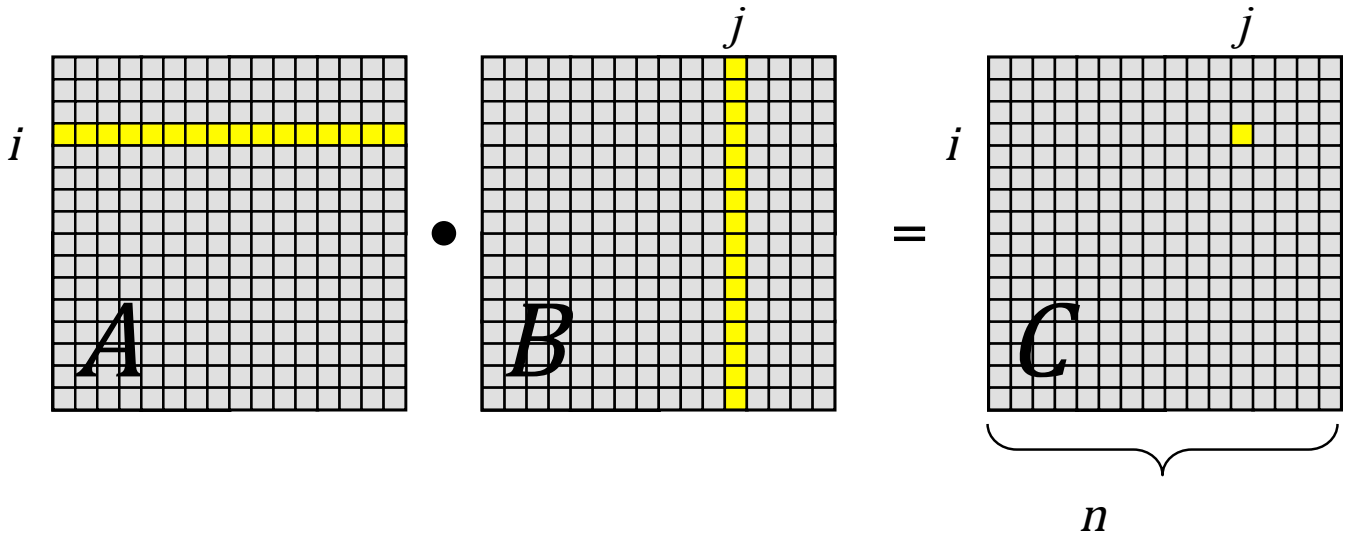
Dissertation Talk
April 17, 2013

Let's start with matrix multiplication

Suppose we want to compute

$$A \cdot B = C$$

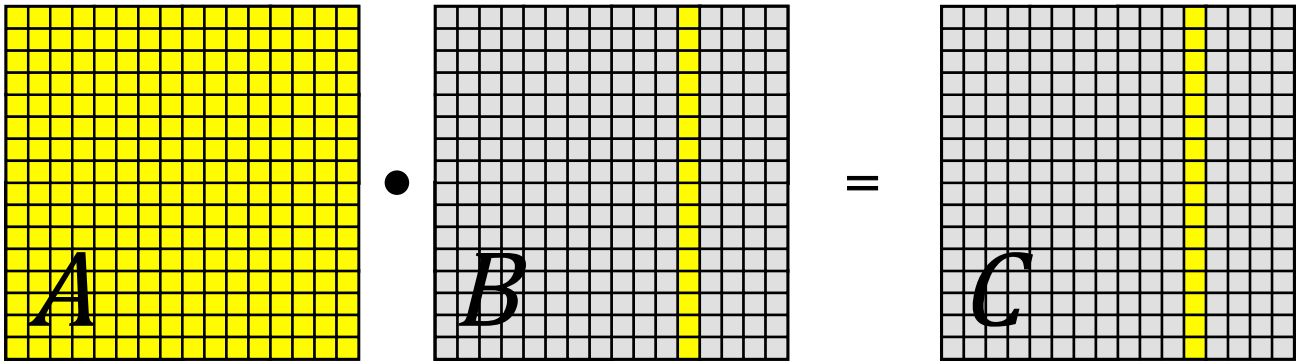
where A and B are $n \times n$ matrices



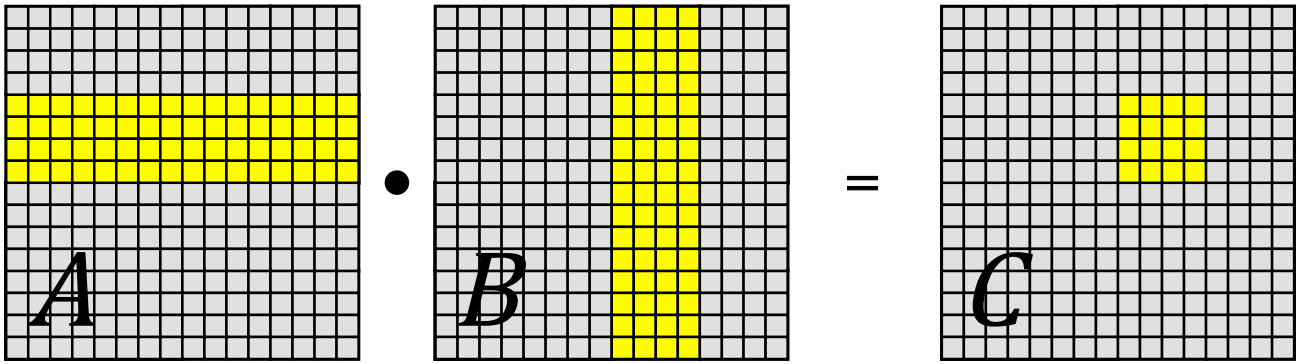
$$\sum_{k=1}^{n} a_{ik} b_{kj} = c_{ij}$$

Two algorithms for matrix multiplication...

We can multiply matrices like this ("matrix-vector" algorithm):

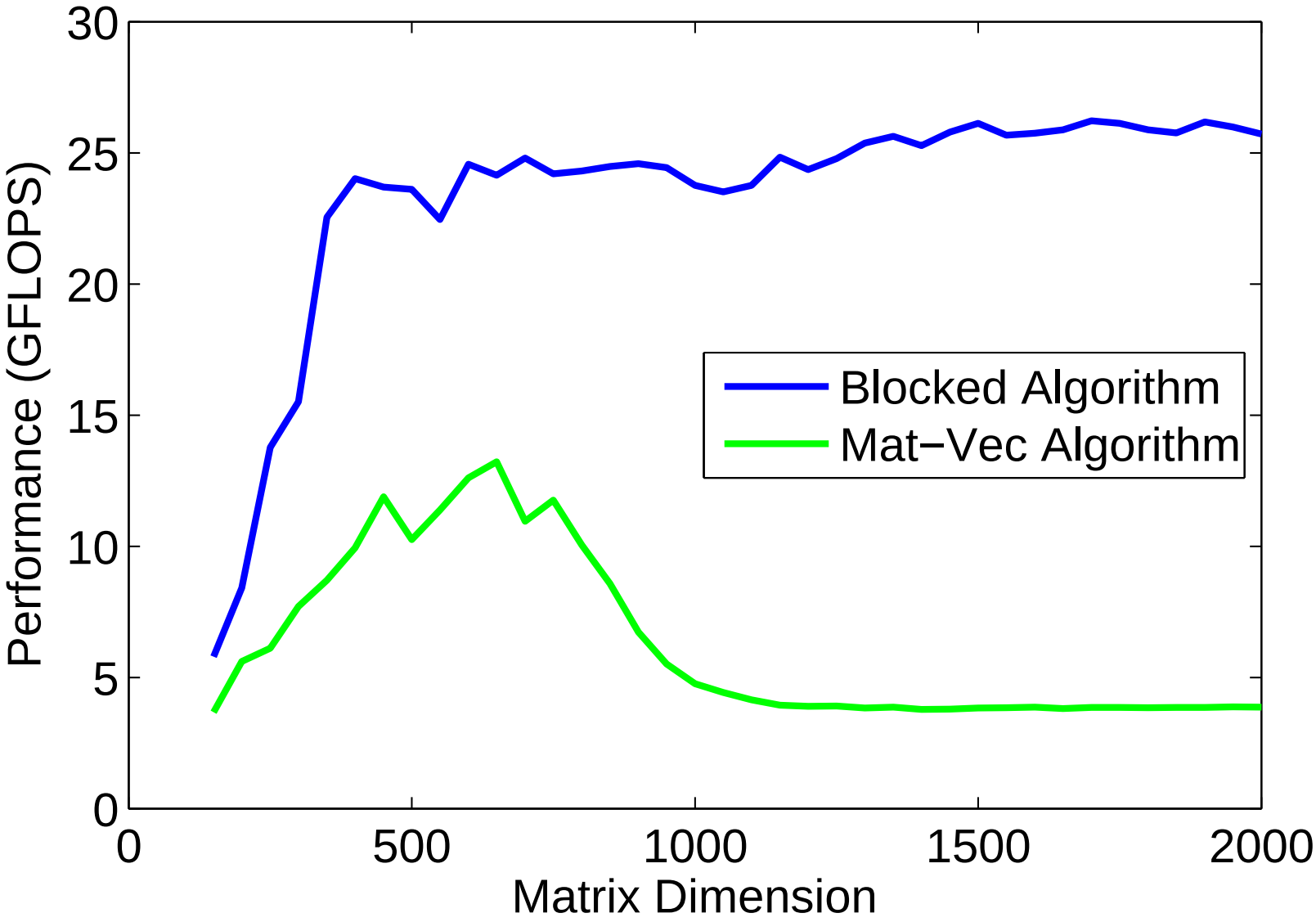


or like this ("blocked" algorithm):



In both cases, we do $2n^3 + O(n^2)$ flops

Same computation, different performance

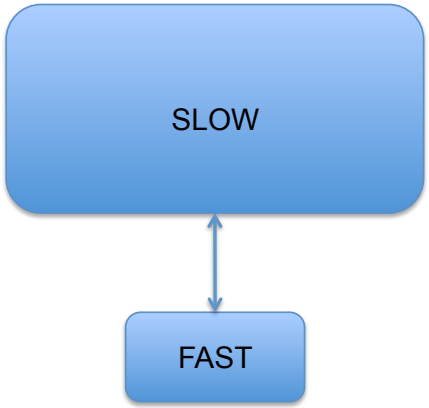


We must consider communication

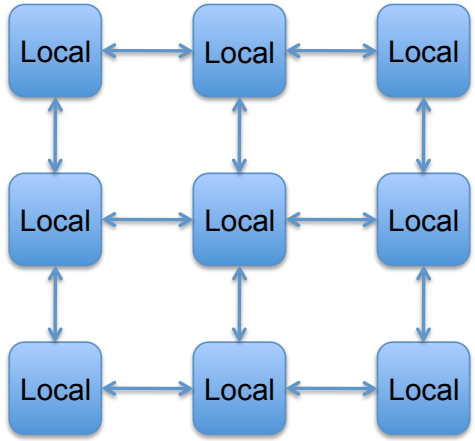
By *communication*, I mean

- moving data within memory hierarchy on a sequential computer
- moving data between processors on a parallel computer

For high-level analysis, we'll use these simple memory models:



Sequential



Parallel

Runtime model

Measure computation in terms of *# flops* performed

Time per flop: γ

Measure communication in terms of *# words* communicated

Time per word: β

Total running time of an algorithm (ignoring overlap):

$$\gamma \cdot (\text{\# flops}) + \beta \cdot (\text{\# words})$$

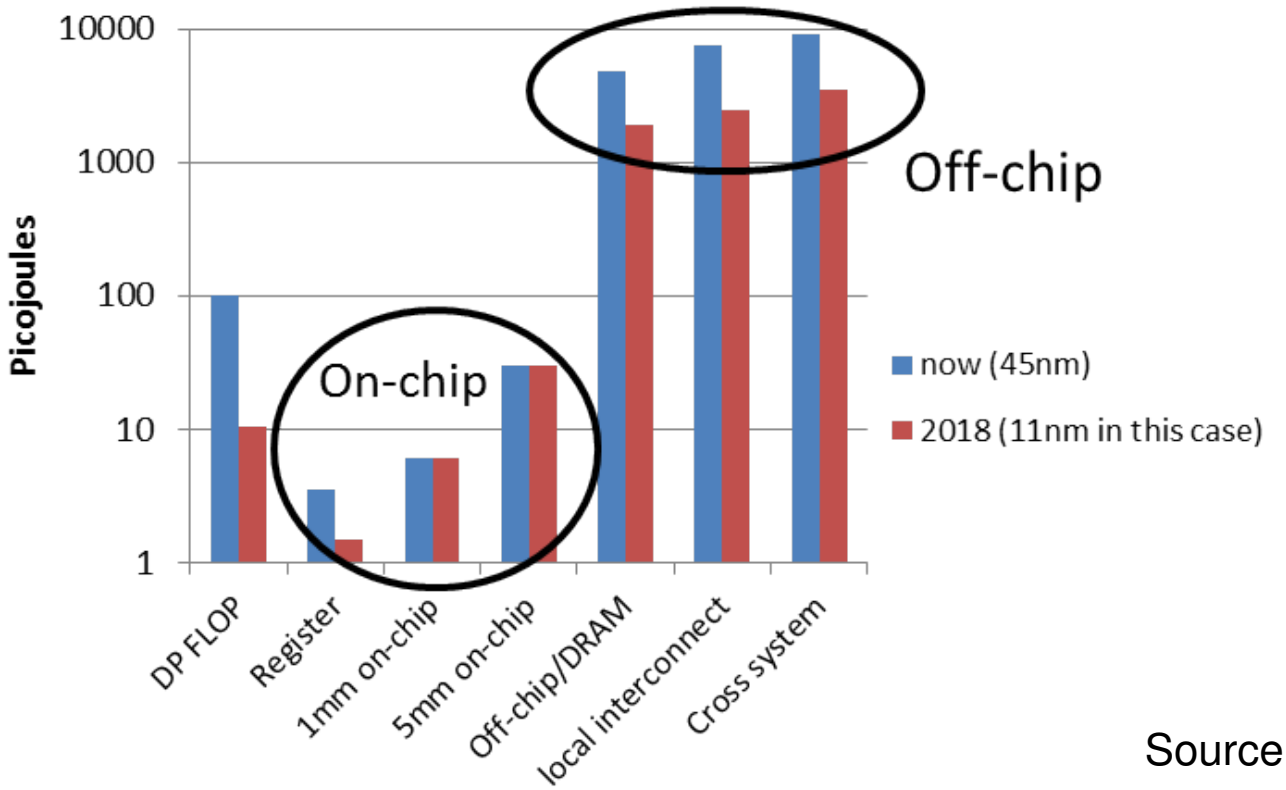
$\beta \gg \gamma$ as measured in time *and* energy, and the relative cost of communication is increasing

Why avoid communication

Annual Improvements in Time

Flop rate γ	DRAM Bandwidth β	Network Bandwidth β
59% per year	23% per year	26% per year

Energy cost comparisons

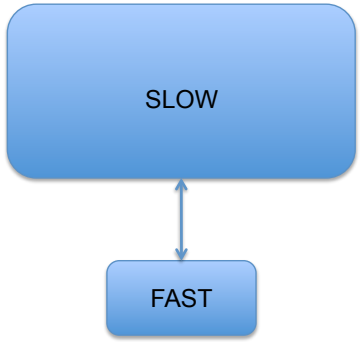


Source: John Shalf

Costs of matrix multiplication algorithms

Let M be the size of the fast memory

The blocked algorithm uses a block size of $\sqrt{M/3}$

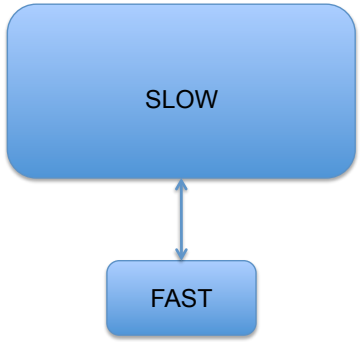


	Computation (# flops)	**Communication (# words)**
Mat-Vec Algorithm	$O(n^3)$	$O(n^3)$
Blocked Algorithm	$O(n^3)$	$O\left(\frac{n^3}{\sqrt{M}}\right)$

Costs of matrix multiplication algorithms

Let M be the size of the fast memory

The blocked algorithm uses a block size of $\sqrt{M/3}$



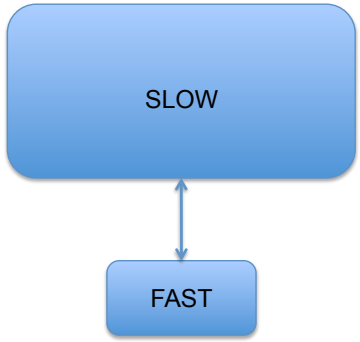
	Computation (# flops)	**Communication (# words)**
Mat-Vec Algorithm	$O(n^3)$	$O(n^3)$
Blocked Algorithm	$O(n^3)$	$O\left(\frac{n^3}{\sqrt{M}}\right)$

Can we do better than the blocked algorithm?

Costs of matrix multiplication algorithms

Let M be the size of the fast memory

The blocked algorithm uses a block size of $\sqrt{M/3}$



	Computation (# flops)	**Communication (# words)**
Mat-Vec Algorithm	$O(n^3)$	$O(n^3)$
Blocked Algorithm	$O(n^3)$	$O\left(\frac{n^3}{\sqrt{M}}\right)$

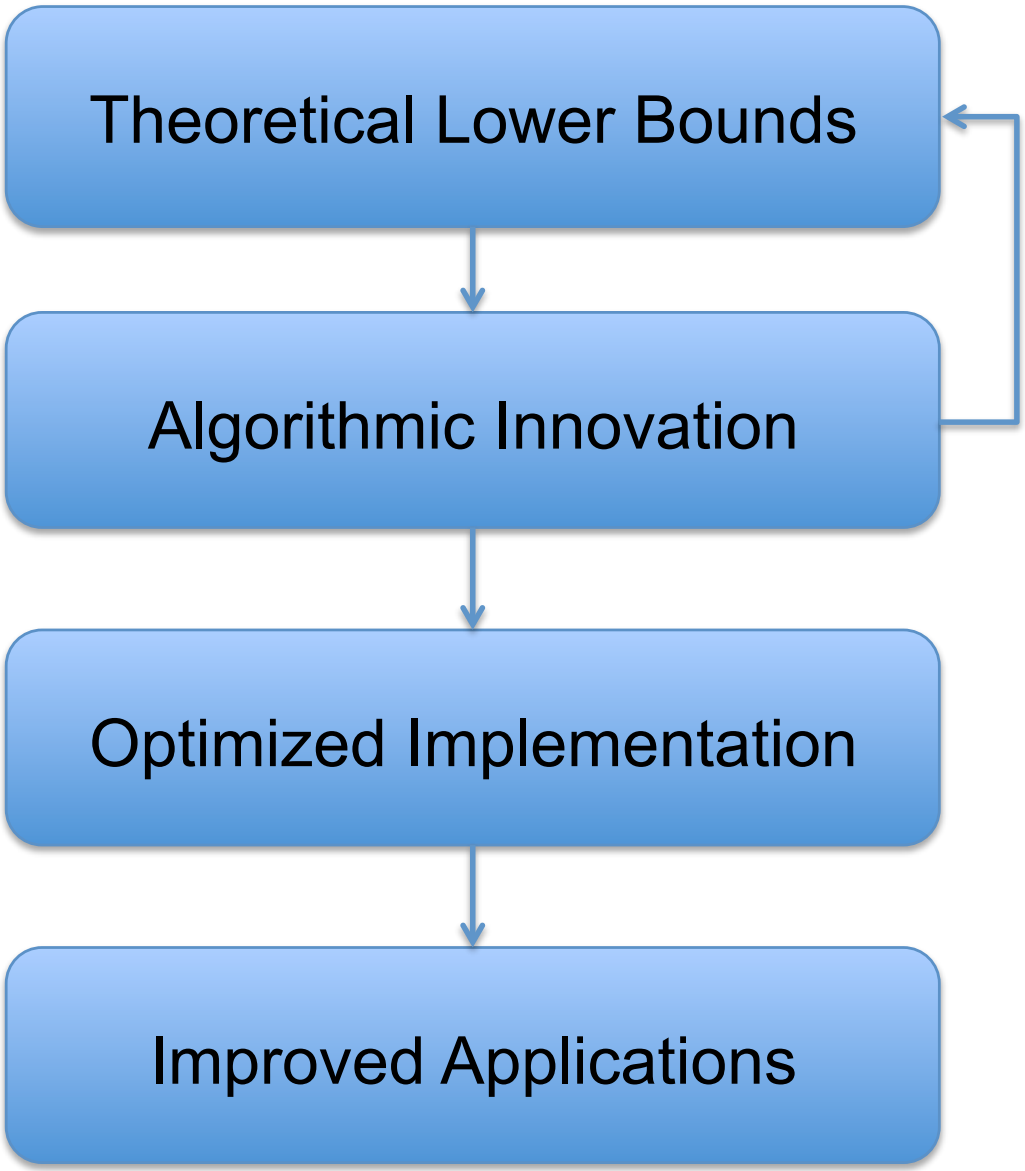
Can we do better than the blocked algorithm?

No . . . and Yes

Summary

- some communication is necessary: we can prove **lower bounds**
  - for Strassen's matrix multiplication*
  - for "classical" dense linear algebra
- theoretical analysis identifies sub-optimal algorithms and spurs **algorithmic innovation**
  - parallel implementation of Strassen's matrix multiplication*
  - solving symmetric indefinite linear system*
  - computing eigenvalues of a symmetric band matrix
  - computing a tall-skinny SVD
  - LU and QR factorizations
  - nonsymmetric eigendecompositions
- minimizing communication leads to speedups in practice

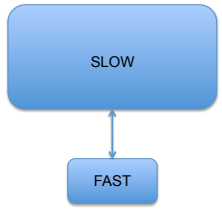
Theory to practice



Lower bounds for classical matrix multiplication

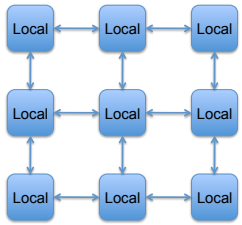
- Assume $O(n^3)$ algorithm
- Sequential case with fast memory of size M
  - lower bound on words moved between fast/slow mem:

$$\Omega\left(\frac{n^3}{\sqrt{M}}\right) \quad \text{[Hong \& Kung 81]}$$



  - attained by blocked algorithm
- Parallel case with P processors (local memory of size M)
  - lower bound on words communicated (along critical path):

$$\Omega\left(\frac{n^3}{P\sqrt{M}}\right) \quad \text{[Toledo et al. 04]}$$



  - also attainable

Let's ask again:

Can we do better than the blocked algorithm?

Given the computation involved, it minimized communication. . .

Let's ask again:

Can we do better than the blocked algorithm?

Given the computation involved, it minimized communication. . .

. . . but what if we change the computation?

It's possible to reduce both computation *and* communication

Strassen's algorithm for matrix multiplication

Strassen showed how to use 7 multiplies instead of 8 for 2×2 multiplication

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Classical Algorithm

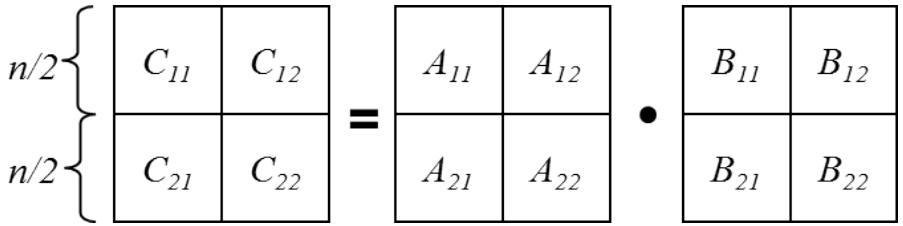
$$\begin{aligned}
M_1 &= A_{11} \cdot B_{11} \\
M_2 &= A_{12} \cdot B_{21} \\
M_3 &= A_{11} \cdot B_{12} \\
M_4 &= A_{12} \cdot B_{22} \\
M_5 &= A_{21} \cdot B_{11} \\
M_6 &= A_{22} \cdot B_{21} \\
M_7 &= A_{21} \cdot B_{12} \\
M_8 &= A_{22} \cdot B_{22} \\
C_{11} &= M_1 + M_2 \\
C_{12} &= M_3 + M_4 \\
C_{21} &= M_5 + M_6 \\
C_{22} &= M_7 + M_8
\end{aligned}$$

Strassen's Algorithm

$$\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) \\
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}$$

Strassen's algorithm for matrix multiplication

Strassen showed how to use 7 multiplies instead of 8 for 2×2 multiplication



$$
\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\end{aligned}
$$

Flop count recurrence:

$$F(n) = 7 \cdot F(n/2) + \Theta(n^2)$$

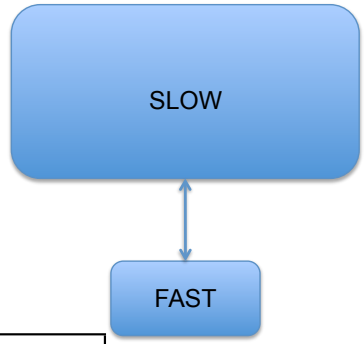
$$F(n) = \Theta\left(n^{\log_2 7}\right)$$

$$\log_2 7 \approx 2.81$$

$$
\begin{aligned}
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}
$$

Sequential communication costs

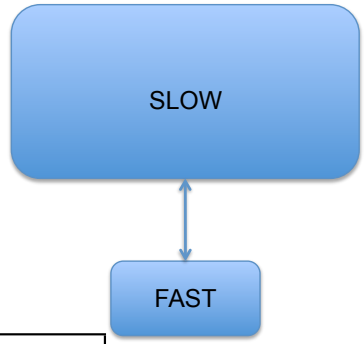
If you implement Strassen's algorithm recursively on a sequential computer:



	Computation # flops	**Communication # words**
Classical (blocked)	$O(n^3)$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^3 M\right)$
Strassen	$O(n^{\log_2 7})$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$

Sequential communication costs

If you implement Strassen's algorithm recursively on a sequential computer:



	Computation # flops	**Communication # words**
Classical (blocked)	$O(n^3)$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^3 M\right)$
Strassen	$O(n^{\log_2 7})$	$O\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$

Can we reduce Strassen's communication cost further?

Lower bounds for Strassen's algorithm

Theorem (Ballard, Demmel, Holtz, Schwartz 12)

On a sequential machine, Strassen's algorithm must communicate

$$\#\ words = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$$

and on a parallel machine, it must communicate

$$\#\ words = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} \frac{M}{P}\right)$$

Lower bounds for Strassen's algorithm

Theorem (Ballard, Demmel, Holtz, Schwartz 12)

On a sequential machine, Strassen's algorithm must communicate

$$\#\ words = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} M\right)$$

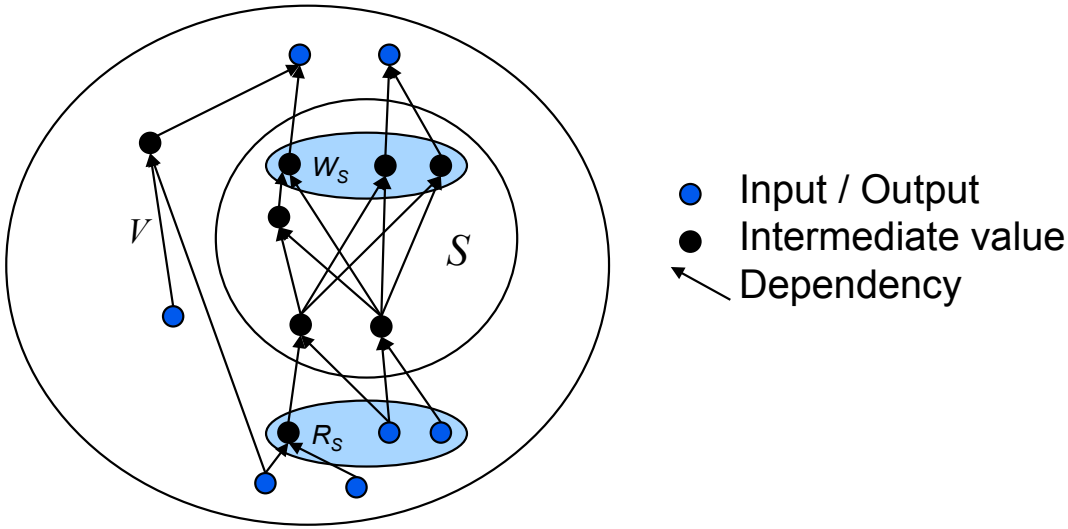
and on a parallel machine, it must communicate

$$\#\ words = \Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7} \frac{M}{P}\right)$$

This work

- received the *SPAA Best Paper Award* [BDHS11]
- appeared in the *Journal of the ACM* [BDHS12a]
- and has been invited to appear as a Research Highlight in the *Communications of the ACM*

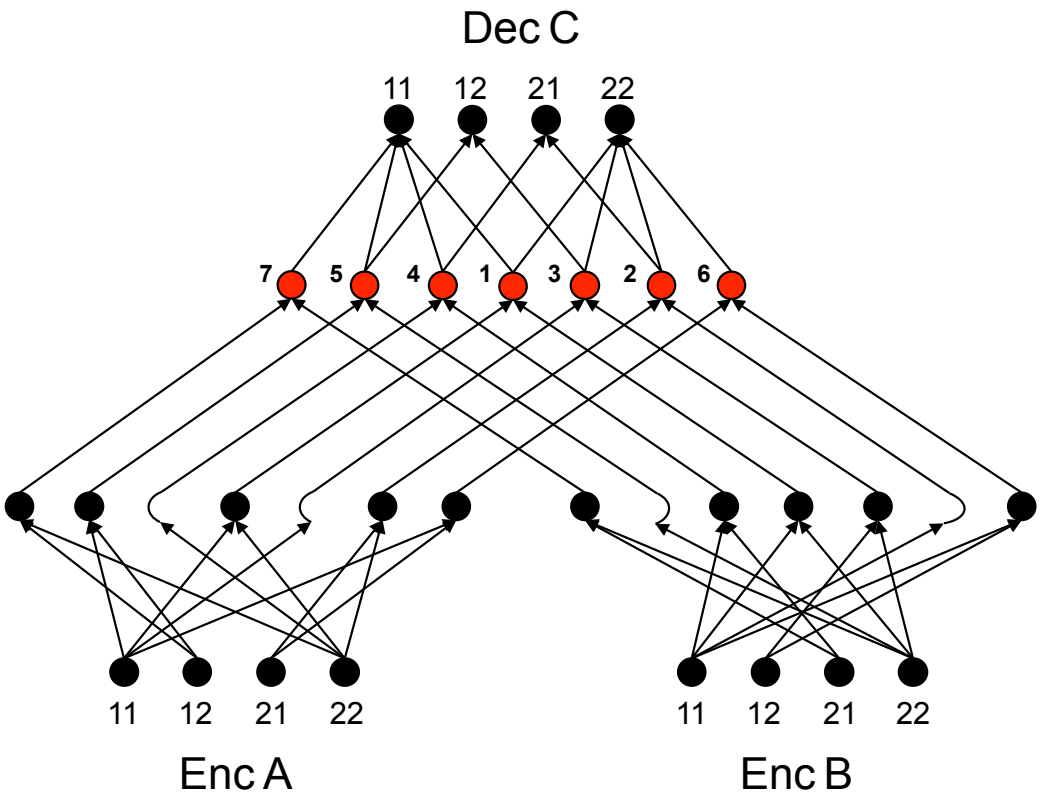
Computation graph analysis



We connected graph *expansion* to communication

- expansion describes the relationship between a subset and its neighbors in the complement
- larger expansion implies more communication necessary

Strassen's computation graph



$$
\begin{aligned}
M_1 &= (A_{11} + A_{22}) \cdot (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \cdot B_{11} \\
M_3 &= A_{11} \cdot (B_{12} - B_{22}) \\
M_4 &= A_{22} \cdot (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \cdot B_{22} \\
M_6 &= (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) \\
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}
$$

Optimal Parallel Algorithm?

This lower bound proves that the sequential recursive algorithm is communication-optimal

What about the parallel case?

Optimal Parallel Algorithm?

This lower bound proves that the sequential recursive algorithm is communication-optimal

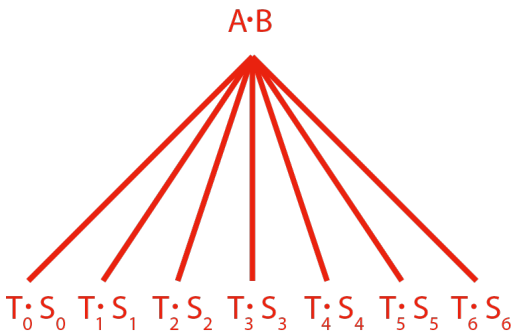
What about the parallel case?

- Earlier attempts to parallelize Strassen had communication costs which exceeded the lower bound
- We developed a new algorithm that is communication-optimal, called Communication-Avoiding Parallel Strassen (CAPS) [BDH+12b]

Main idea of CAPS algorithm

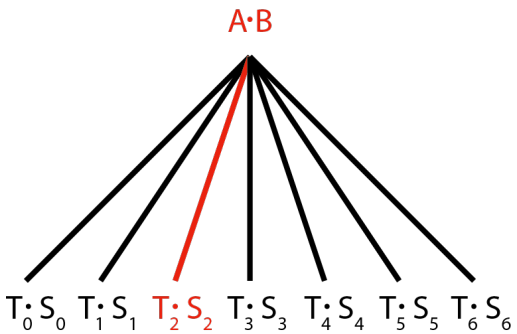
At each level of recursion tree, choose either breadth-first or depth-first traversal of the recursion tree

Breadth-First-Search (BFS)



- Runs all 7 multiplies in parallel
  - each uses $P/7$ processors
- Requires $7/4$ as much extra memory
- Requires communication, but minimizes communication in subtrees

Depth-First-Search (DFS)



- Runs all 7 multiplies sequentially
  - each uses all P processors
- Requires $1/4$ as much extra memory
- Increases communication by factor of $7/4$ in subtrees

Is it optimal?

After algorithmic analysis, we can compare communication costs to the lower bound:

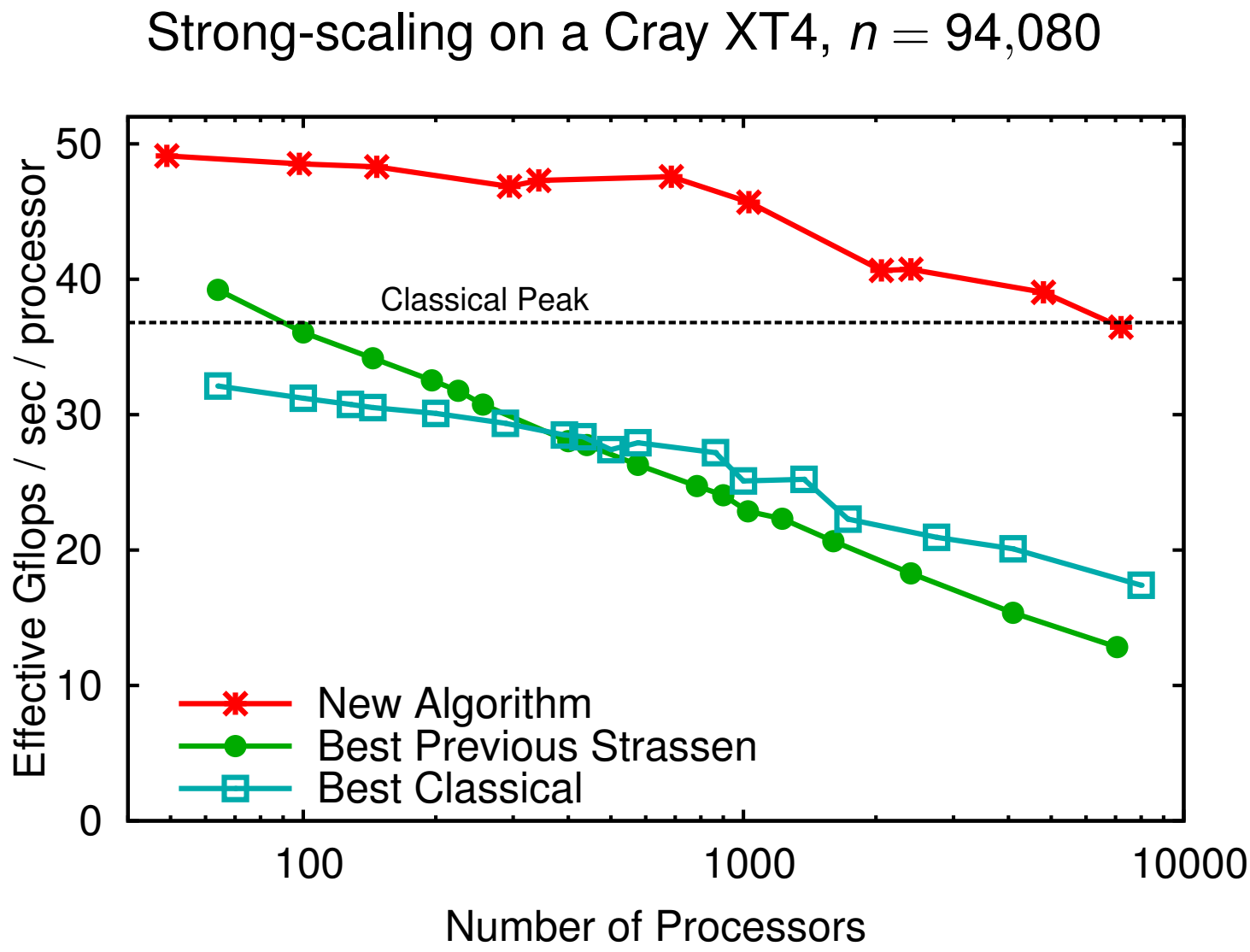
	Communication # words
CAPS	$O\left(\max\left\{\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7}\frac{M}{P}, \frac{n^2}{P^{2/\log_2 7}}\right\}\right)$
Lower Bound [BDHS11]	$\Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7}\frac{M}{P}\right)$

Is it optimal?

After algorithmic analysis, we can compare communication costs to the lower bound:

	Communication # words
CAPS	$O\left(\max\left\{\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7}\frac{M}{P}, \frac{n^2}{P^{2/\log_2 7}}\right\}\right)$
Lower Bound [BDHS11]	$\Omega\left(\left(\frac{n}{\sqrt{M}}\right)^{\log_2 7}\frac{M}{P}\right)$
New Lower Bound [BDH+12a]	$\Omega\left(\frac{n^2}{P^{2/\log_2 7}}\right)$

Performance of CAPS on a large problem



- Actual Peak
- Strassen-Winograd peak
- Performance Model

Can we beat Strassen?

Strassen's algorithm allows for less computation and communication than the classical $O(n^3)$ algorithm

We have algorithms that attain its communication lower bounds and perform well on highly parallel machines

Can we do any better?

Can we beat Strassen?

Strassen's algorithm allows for less computation and communication than the classical $O(n^3)$ algorithm

We have algorithms that attain its communication lower bounds and perform well on highly parallel machines

Can we do any better?

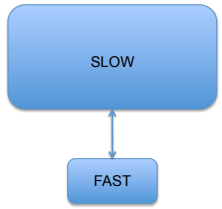
Yes, but there are other complications

Let's go back to classical matrix algorithms

Lower bounds for classical matrix multiplication

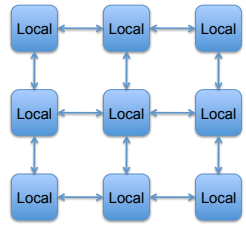
- Assume $O(n^3)$ algorithm
- Sequential case with fast memory of size M
  - lower bound on words moved between fast/slow mem:

$$\Omega\left(\frac{n^3}{\sqrt{M}}\right) \quad \text{[Hong \& Kung 81]}$$



  - attained by blocked algorithm
- Parallel case with P processors (local memory of size M)
  - lower bound on words communicated (along critical path):

$$\Omega\left(\frac{n^3}{P\sqrt{M}}\right) \quad \text{[Toledo et al. 04]}$$



  - also attainable

Extensions to the rest of linear algebra

Theorem (Ballard, Demmel, Holtz, Schwartz 11)

If a computation "smells" like 3 nested loops, it must communicate

$$\#\ words = \Omega\left(\frac{\#\ flops}{\sqrt{memory\ size}}\right)$$

This result applies to

- dense or sparse problems
- sequential or parallel computers

This work was recognized with the *SIAM Linear Algebra Prize*, given to the best paper from the years 2009-2011

▸ Main Idea

Extensions to the rest of linear algebra

Theorem (Ballard, Demmel, Holtz, Schwartz 11)

If a computation "smells" like 3 nested loops, it must communicate

$$\#\ words = \Omega\left(\frac{\#\ flops}{\sqrt{memory\ size}}\right)$$

What smells like 3 nested loops?

- the rest of BLAS 3 (e.g. matrix multiplication, triangular solve)
- Cholesky, LU, LDL^T, LTL^T decompositions
- QR decomposition
- eigenvalue and SVD reductions
- sequences of algorithms (e.g. repeated matrix squaring)
- graph algorithms (e.g. all pairs shortest paths)

This work was recognized with the *SIAM Linear Algebra Prize*, given to the best paper from the years 2009-2011

▸ Main Idea

Extensions to the rest of linear algebra

Theorem (Ballard, Demmel, Holtz, Schwartz 11)

If a computation "smells" like 3 nested loops, it must communicate

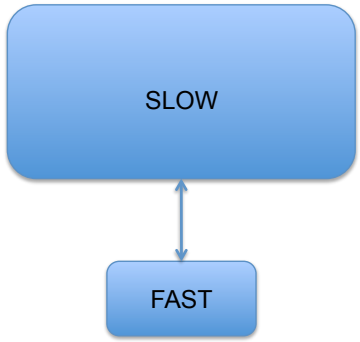
$$\#\ words = \Omega\left(\frac{\#\ flops}{\sqrt{memory\ size}}\right)$$

What if the computation smells like 5 nested loops?

. . . come see Nick's talk next week

▸ Main Idea

Optimal algorithms - sequential $O(n^3)$ linear algebra



Computation	Optimal Algorithm
BLAS 3	blocked algorithms [Gustavson 97]
Cholesky	LAPACK [Ahmed & Pingali 00] [BDHS10]
Symmetric Indefinite	LAPACK (rarely) [BDD+12a]
LU	LAPACK (rarely) [Toledo 97]* [Grigori et al. 11]
QR	LAPACK (rarely) [Frens & Wise 03] [Elmroth & Gustavson 98]* [Hoemmen et al. 12]*
Eig, SVD	[BDK12a], [BDD12b]

▸ Parallel Case

Example: symmetric indefinite linear solve

Suppose we want to solve $Ax = b$ where A

- is symmetric (save half the storage and flops)
- but indefinite (need to permute rows/cols for numerical stability)

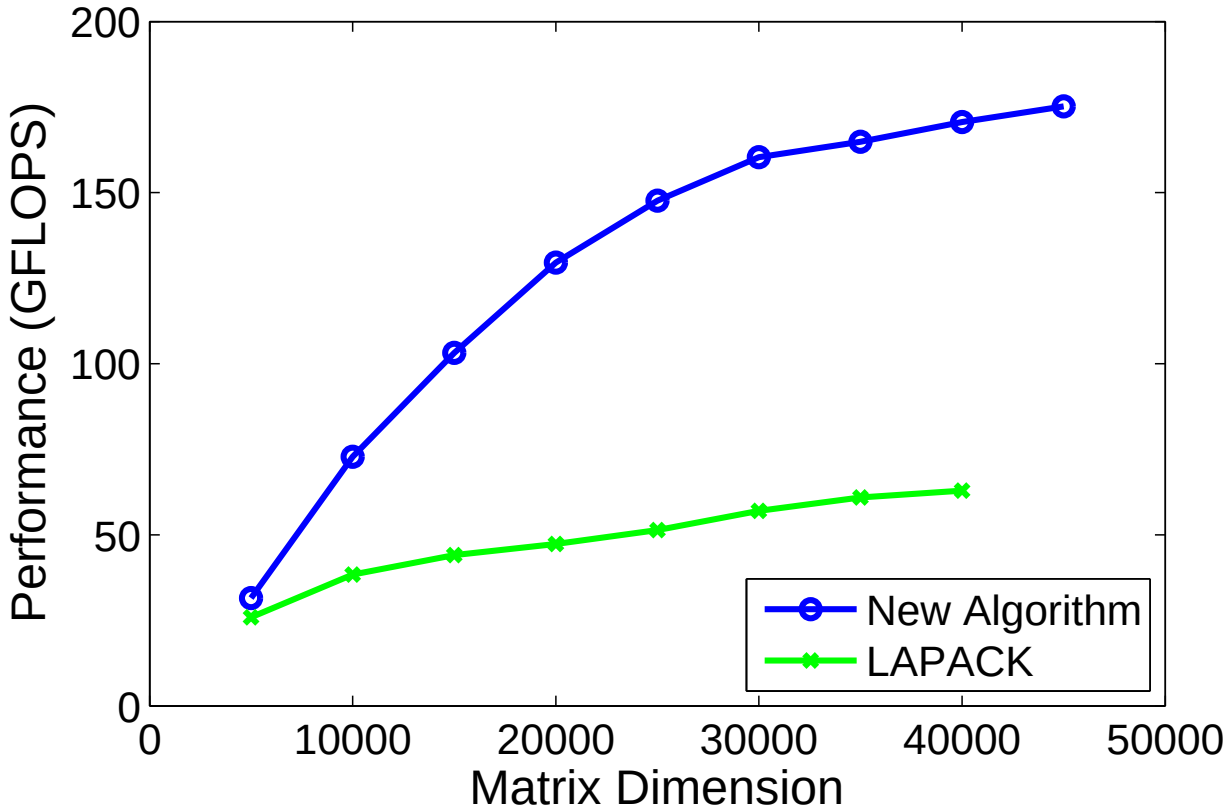
We generally want to compute a factorization

$$PAP^T = LTL^T$$

P is a permutation, L is triangular, and T is symmetric and "simpler"

Reducing communication improves performance

Performance of symmetric indefinite linear system solvers



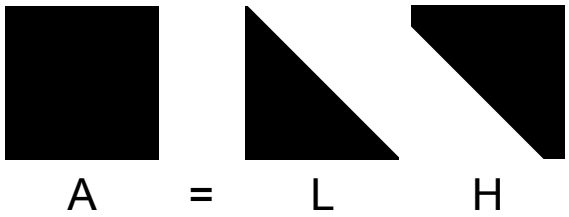
Implemented within PLASMA library [BBD+13]

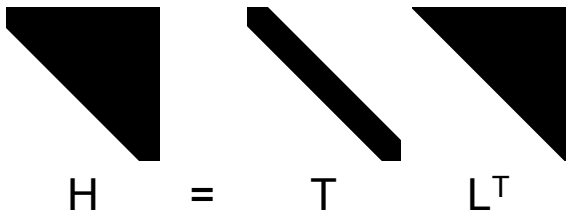
This work will receive a *Best Paper Award* at IPDPS '13

Aasen's symmetric indefinite factorization

We're solving $Ax = b$ where $A = A^T$ but A is indefinite

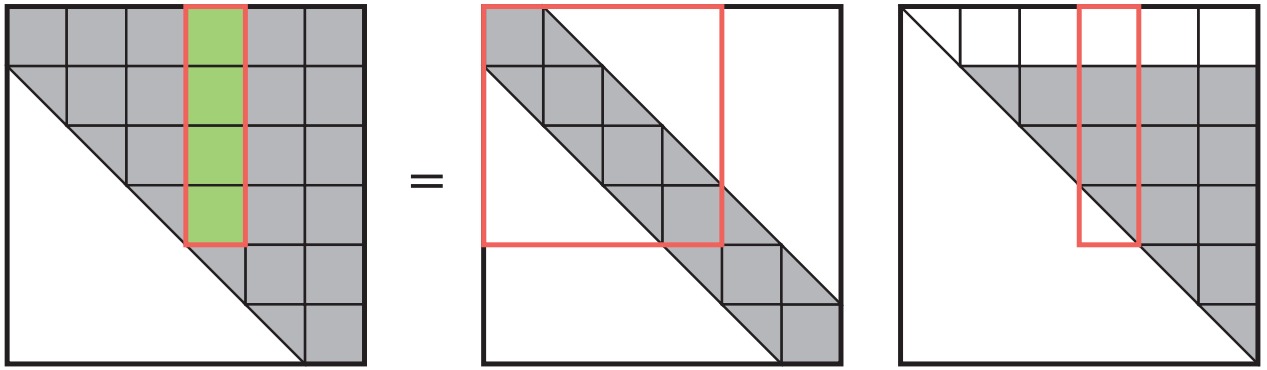
- Standard approach is to compute $PAP^T = LDL^T$
  - L is lower triangular and D is block diagonal (1×1 and 2×2 blocks)
  - requires complicated pivoting, harder to do tournament pivoting
- Aasen's approach is to compute $PAP^T = LTL^T$ [Aas71]
  - L is lower triangular and T is tridiagonal
  - pivoting is more like LU (nonsymmetric case)



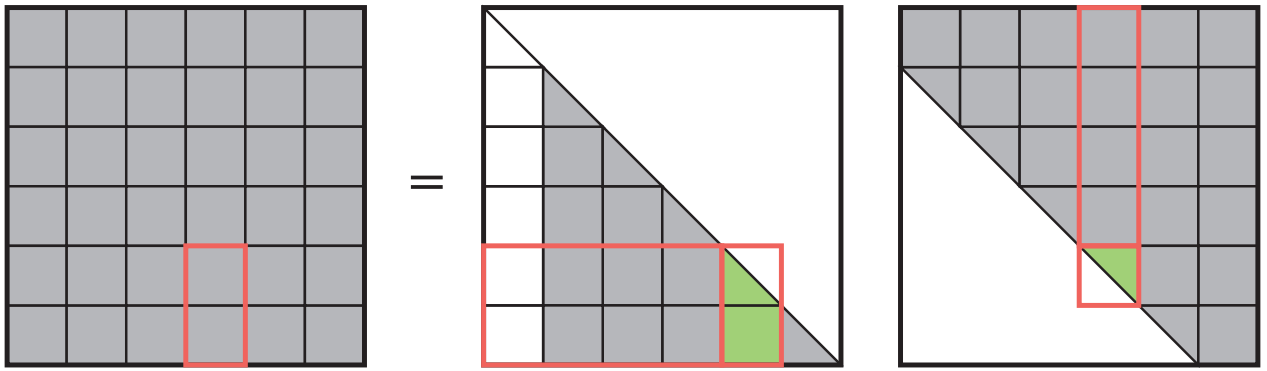


Blocked version of Aasen's algorithm

Compute block column of H from T and L:



Compute block column of L and subdiagonal block of H with LU:

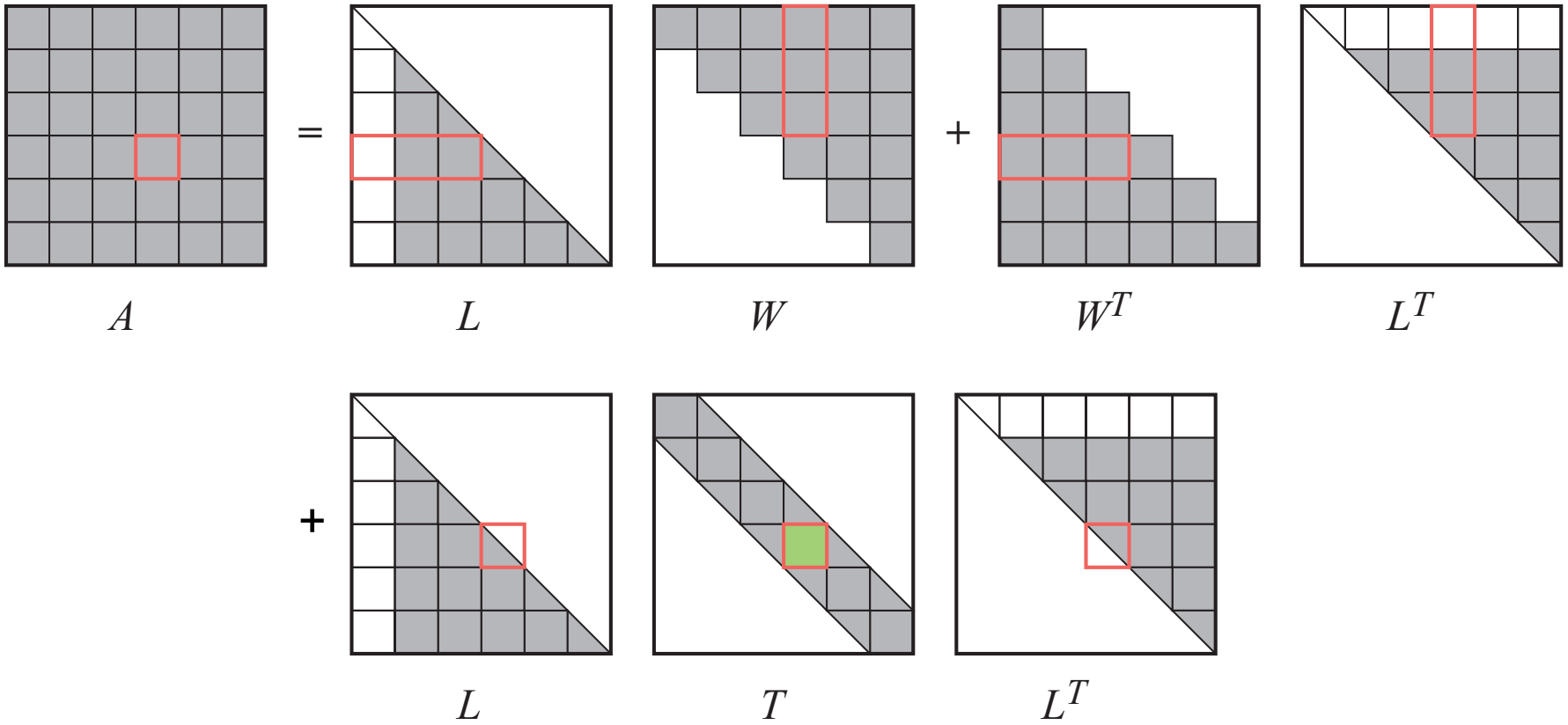


Converting scalar to blocked algorithm...

1. $\underline{H_{1:J-1,J}} = T_{1:J-1,1:J-1} L^T_{1:J-1,J}$
2. $A_{J,J} = L_{J,1:J-1} H_{1:J-1,J} + L_{J,J} \underline{H_{J,J}}$
3. $H_{J,J} = T_{J,J-1} L^T_{J-1,J} + \underline{T_{J,J}} L^T_{J,J}$
4. $A_{J+1:N,J} = L_{J+1:N,1:J} H_{1:J,J} + \underline{L_{J+1:N,J+1} H_{J+1,J}}$
5. $H_{J+1,J} = \underline{T_{J+1,J}} L^T_{J,J}$

Computing symmetric blocks of T

Since diagonal blocks of T are symmetric, need to be computed from a symmetric equation, which includes two-sided triangular solve:



$W \sim H$

Other complications

- How to do tall-skinny LU decomposition?
  - use tournament pivoting
  - use recursive algorithm
  - use LAPACK algorithm
- Need to take care in applying symmetric permutations
- We still need to decompose band matrix T:
  - non-symmetric band LU decomposition
  - successive band reduction (orthogonal similarity transformations)
  - Kaufman's symmetric retraction algorithm

Comm-optimal symmetric indefinite factorization

- After handling all the complications, we obtain a blocked version of Aasen's algorithm which moves

$$O\left(\frac{n^3}{\sqrt{M}}\right) \text{ words}$$

  and matches the communication lower bound

- A shared-memory implementation in the PLASMA library outperforms the best implementation of the standard algorithm

Summary

- some communication is necessary: we can prove **lower bounds**
  - for Strassen's matrix multiplication*
  - for "classical" dense linear algebra
- theoretical analysis identifies sub-optimal algorithms and spurs **algorithmic innovation**
  - parallel implementation of Strassen's matrix multiplication*
  - solving symmetric indefinite linear system*
  - computing eigenvalues of a symmetric band matrix
  - computing a tall-skinny SVD
  - LU and QR factorizations
  - nonsymmetric eigendecompositions
- minimizing communication leads to speedups in practice

Collaborators

- Michael Anderson (UC Berkeley)
- Aydin Buluc (LBNL)
- James Demmel (UC Berkeley)
- Alex Druinsky (Tel-Aviv U)
- Ioana Dumitriu (U Washington)
- Andrew Gearhart (UC Berkeley)
- Laura Grigori (INRIA)
- Olga Holtz (UC Berkeley/TU Berlin)
- Mathias Jacquelin (INRIA)
- Nicholas Knight (UC Berkeley)
- Kurt Keutzer (UC Berkeley)
- Tamara Kolda (Sandia NL)
- Benjamin Lipshitz (UC Berkeley)
- Inon Peled (Tel-Aviv U)
- Todd Plantenga (Sandia NL)
- Oded Schwartz (UC Berkeley)
- Edgar Solomonik (UC Berkeley)
- Sivan Toledo (Tel-Aviv U)
- Ichitaro Yamazaki (UT Knoxville)

Avoiding Communication in Dense Linear Algebra

Grey Ballard

Thank You!

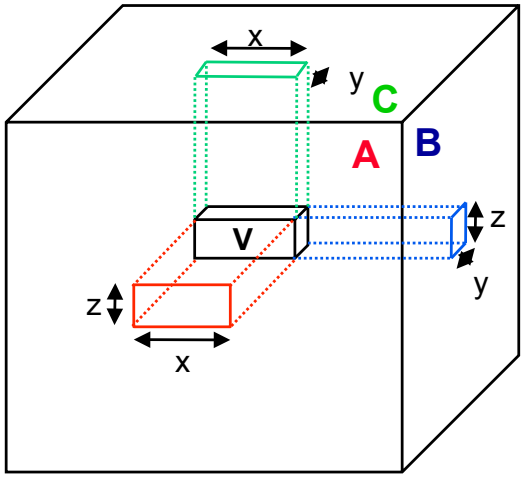
www.eecs.berkeley.edu/~ballard
http://bebop.cs.berkeley.edu

Other Ongoing and Future Projects

- implementing these algorithms in communication-bound settings
  - e.g., SVD of a tall-skinny matrix on a Hadoop cluster
- extending these algorithmic ideas to sparse matrices
  - e.g., sparse matrix-matrix multiplication
- using Strassen to do the rest of linear algebra in parallel
- trading off local memory and communication in parallel QR decomposition

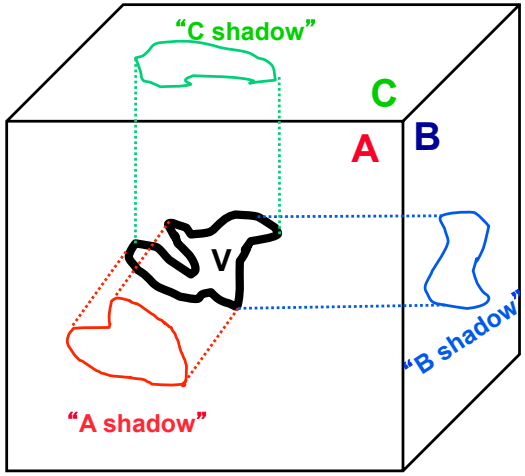
Main Idea of Lower Bound Proof

Crux of proof based on geometric inequality [Loomis & Whitney 49]



Volume of box

$$V = xyz = \sqrt{xz \cdot yz \cdot xy}$$



Volume of a 3D set

$$V \leq \sqrt{\text{area}(\text{A shadow})} \cdot \sqrt{\text{area}(\text{B shadow})} \cdot \sqrt{\text{area}(\text{C shadow})}$$

Given limited set of data, how much useful computation can be done?

Back

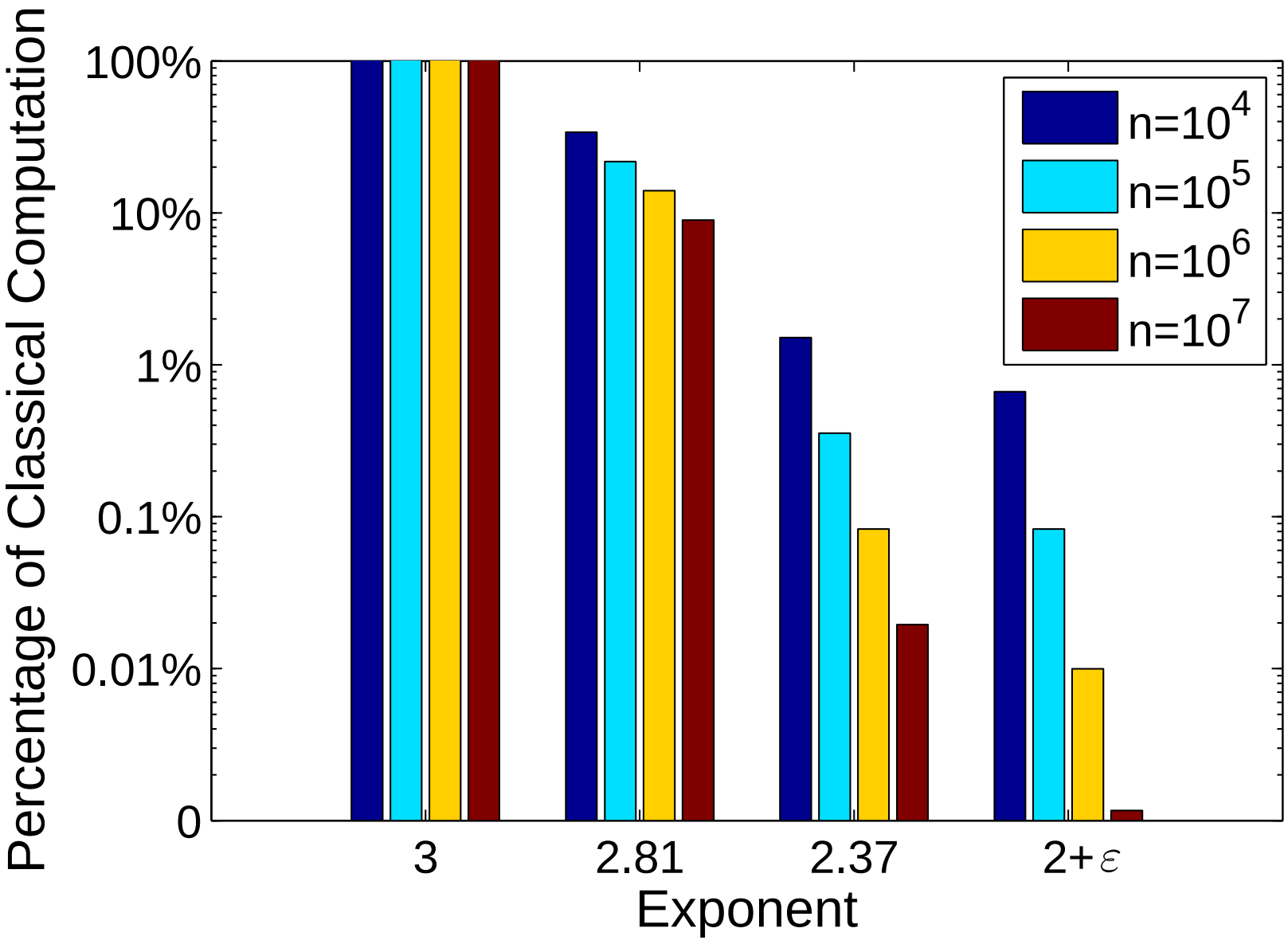
Can an $n \times n$ linear system of equations $Ax = b$ be solved in $O(n^{2+\varepsilon})$ operations, where ε is arbitrarily small?

. . . if solved affirmatively, [this] would change the world.

It is an article of faith for some of us that if $O(n^{2+\varepsilon})$ is ever achieved, the big idea that achieves it will correspond to an algorithm that is really practical.

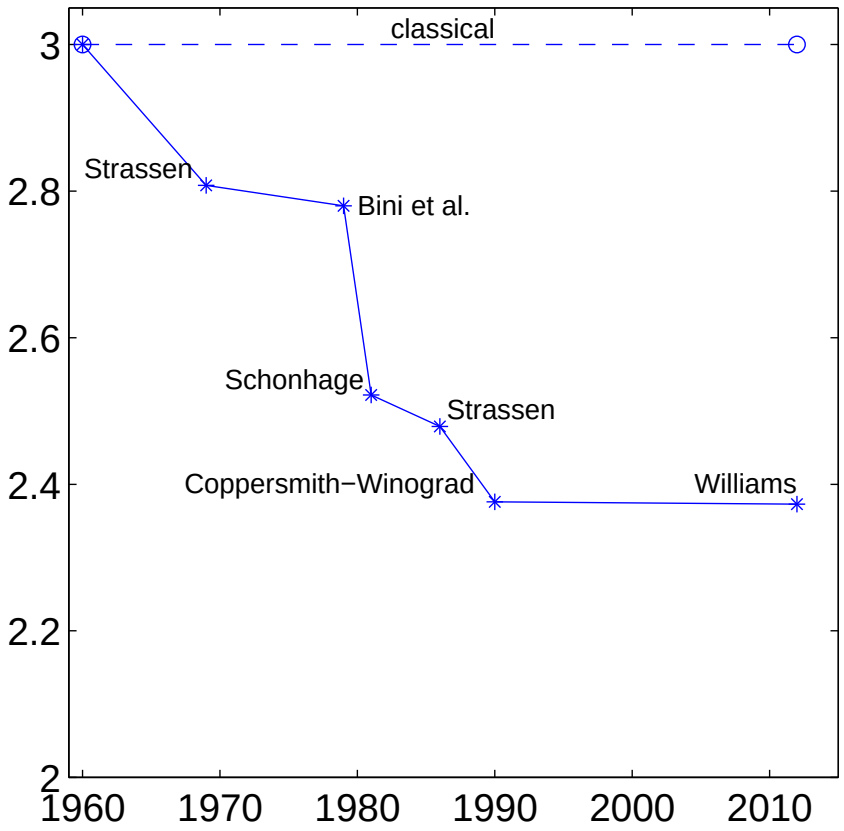
-Nick Trefethen, 2012 SIAM President

How much computation will that save?



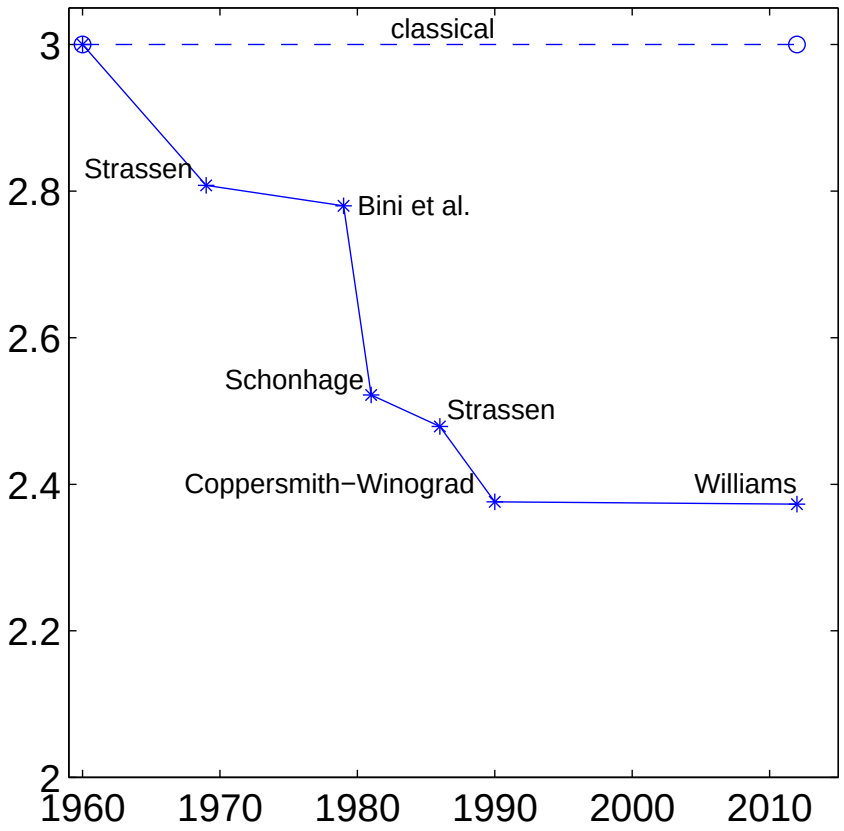
Can we beat Strassen?

Exponent of matrix multiplication over time



Can we beat Strassen?

Exponent of matrix multiplication over time



Unfortunately, these improvements are only theoretical because they

- involve approximations
- are existence proofs
- have (possibly) large constants

Solving the base case...

$2 \times 2 \times 2$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

multiplies	6	7	8
flop count	$O\left(n^{2.58}\right)$	$O\left(n^{2.81}\right)$	$O\left(n^{3}\right)$

Solving the base case...

$2 \times 2 \times 2$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

multiplies	~~6~~	7	8
flop count	~~$O\left(n^{2.58}\right)$~~	$O\left(n^{2.81}\right)$	$O\left(n^{3}\right)$

Solving the base case...

$2 \times 2 \times 2$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

multiplies	~~6~~	7	8
flop count	~~$O\left(n^{2.58}\right)$~~	$O\left(n^{2.81}\right)$	$O\left(n^{3}\right)$

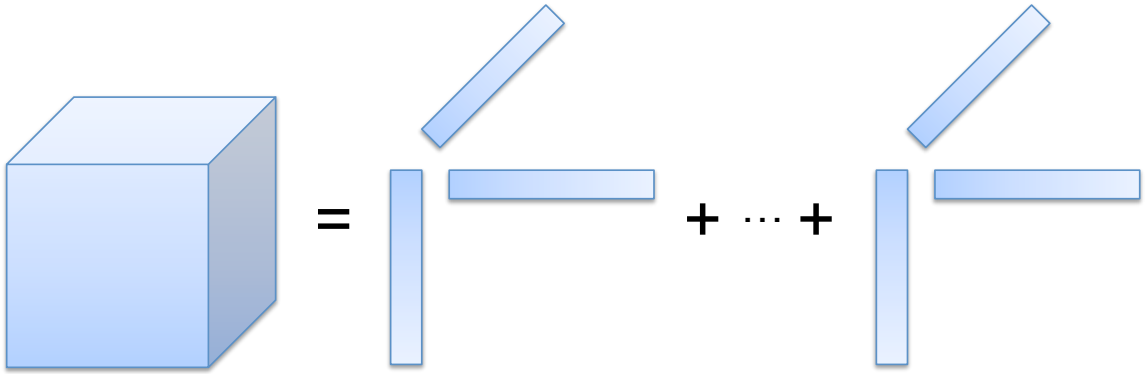
$3 \times 3 \times 3$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

multiplies	19	21	23	27
flop count	$O\left(n^{2.68}\right)$	$O\left(n^{2.77}\right)$	$O\left(n^{2.85}\right)$	$O\left(n^{3}\right)$

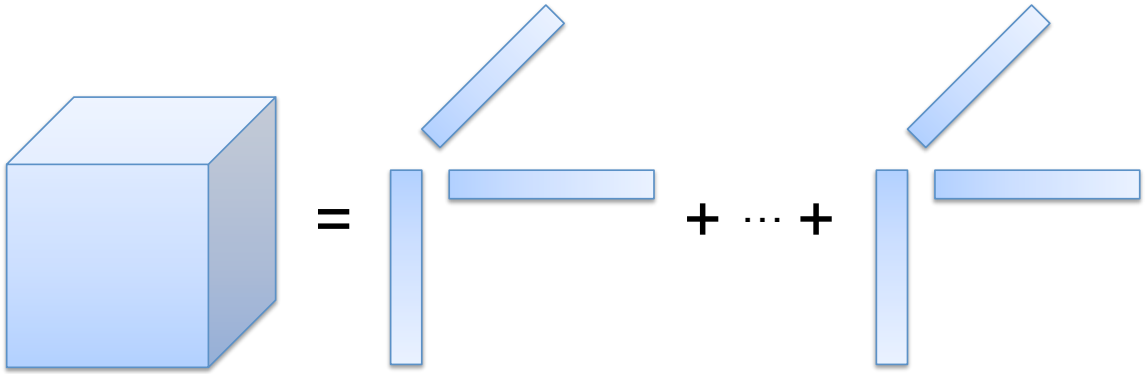
Beating Strassen

Finding a better base case corresponds to computing a low-rank decomposition of a particular 3D tensor



Beating Strassen

Finding a better base case corresponds to computing a low-rank decomposition of a particular 3D tensor



Unfortunately, this is a nonlinear integer optimization problem

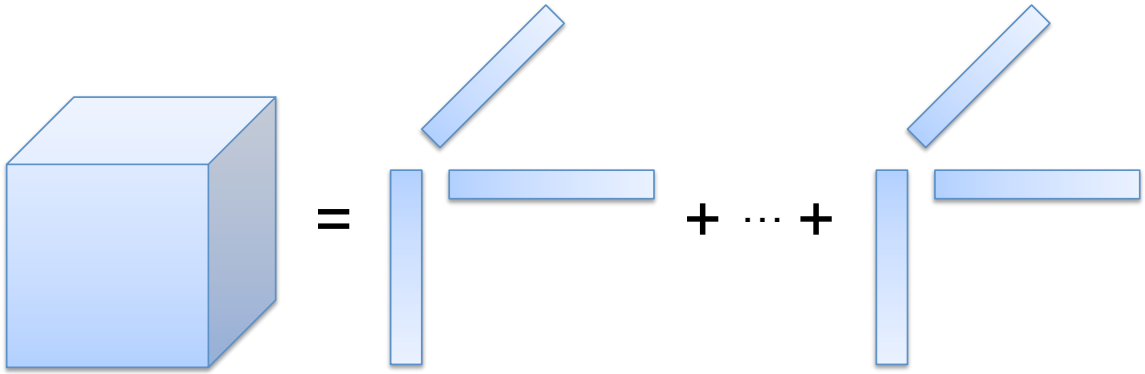
- it's NP-complete (in general), but need to solve it only once
- I used this method to re-discover Strassen

Could use (numerical) low-rank tensor approximation algorithms

- very efficient, but no guarantees

Beating Strassen

Finding a better base case corresponds to computing a low-rank decomposition of a particular 3D tensor



If we find it, we can make it practical!

- same parallelization as Strassen, but with less computation *and* communication

Memory-Independent Lower Bounds

	Classical	Strassen
Memory-dependent lower bound	$\Omega\left(\frac{n^3}{P\sqrt{M}}\right)$	$\Omega\left(\frac{n^\omega}{PM^{\omega/2-1}}\right)$
Memory-independent lower bound	$\Omega\left(\frac{n^2}{P^{2/3}}\right)$	$\Omega\left(\frac{n^2}{P^{2/\omega}}\right)$
Perfect strong scaling range	$P = O\left(\frac{n^3}{M^{3/2}}\right)$	$P = O\left(\frac{n^\omega}{M^{\omega/2}}\right)$
Attaining algorithm	[SD11]	[BDH+12b]

Example: Compute Eigenvalues of Band Matrix

Suppose we want to solve $Ax = \lambda x$ where A

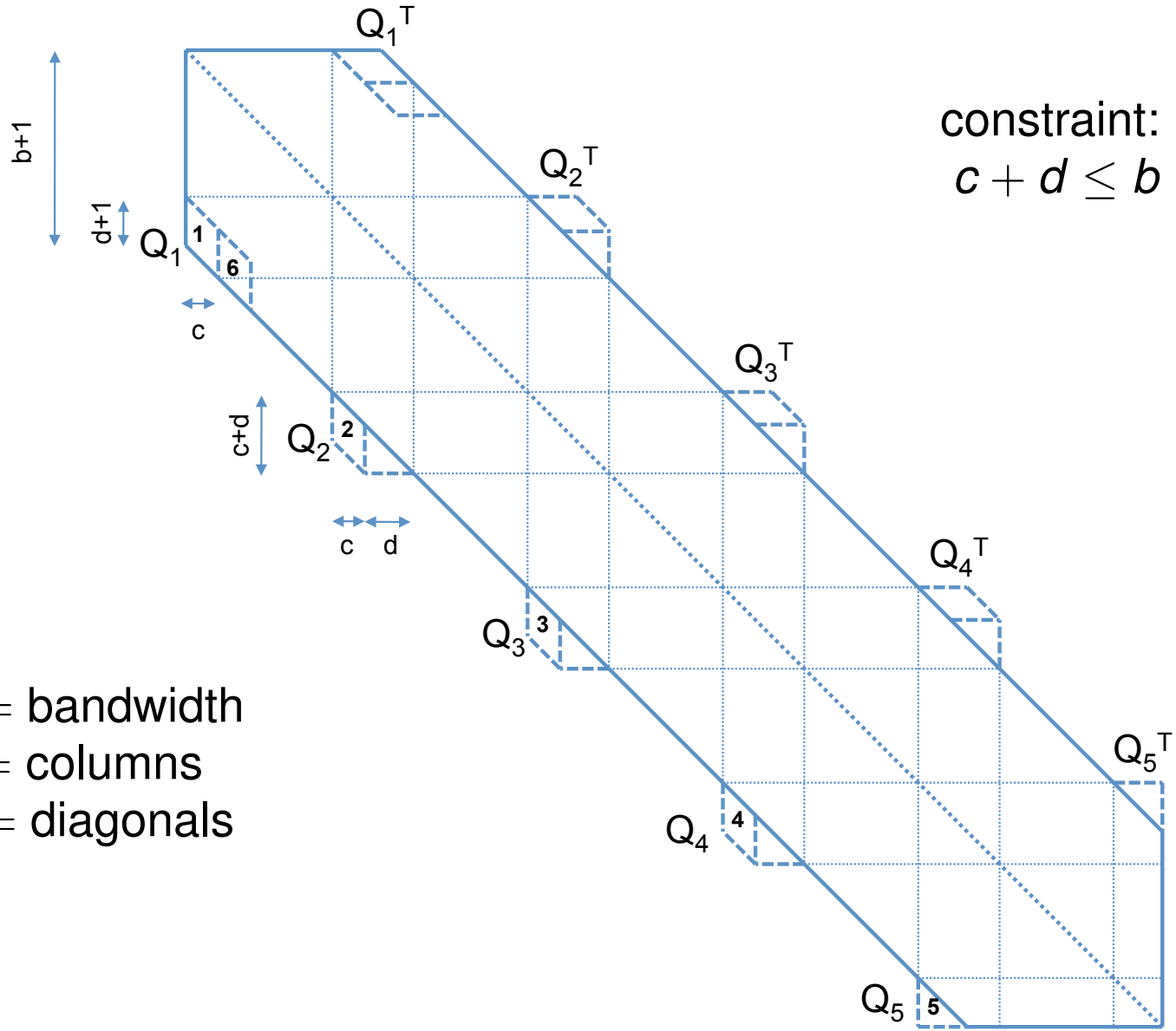
- is symmetric (save half the storage and flops)
- has band structure (exploit sparsity – ignore zeros)

We generally want to compute a factorization

$$A = QTQ^T$$

Q is an orthogonal matrix and T is symmetric tridiagonal

Successive Band Reduction (bulge-chasing)



constraint:
$c + d \leq b$

$b =$ bandwidth
$c =$ columns
$d =$ diagonals

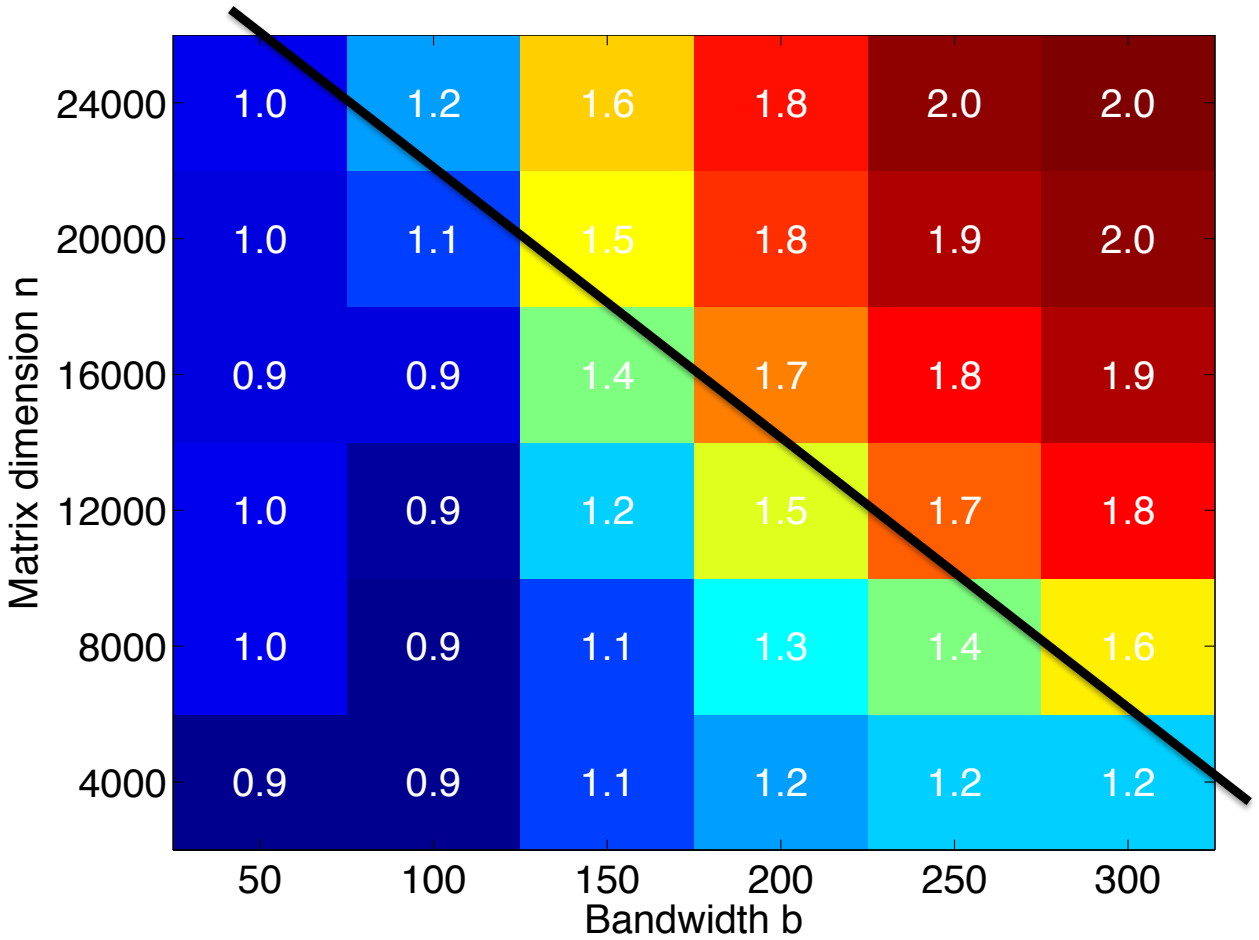
CASBR Data Access Pattern

One bulge at a time

Four bulges at a time

Implementation of Band Eigensolver (CASBR)

Speedup of sequential CASBR over Intel's Math Kernel Library

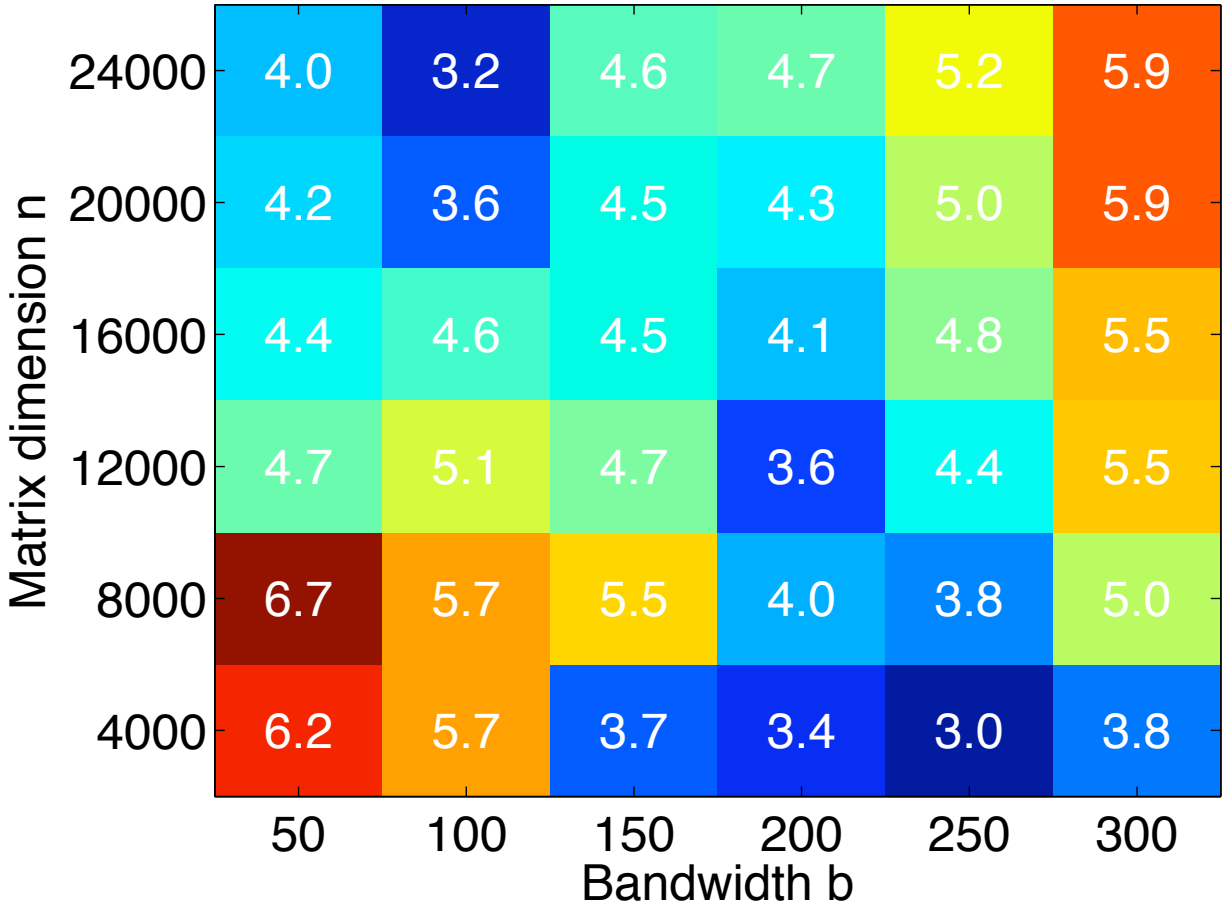


Benchmarked on Intel 10-core Westmere-EX socket [BDK12a]

▸ Algorithmic Details

Implementation of Band Eigensolver (CASBR)

Speedup of parallel CASBR (10 threads) over PLASMA library



Benchmarked on Intel 10-core Westmere-EX socket [BDK12a]

▸ Algorithmic Details

Example Application: Video Background Subtraction

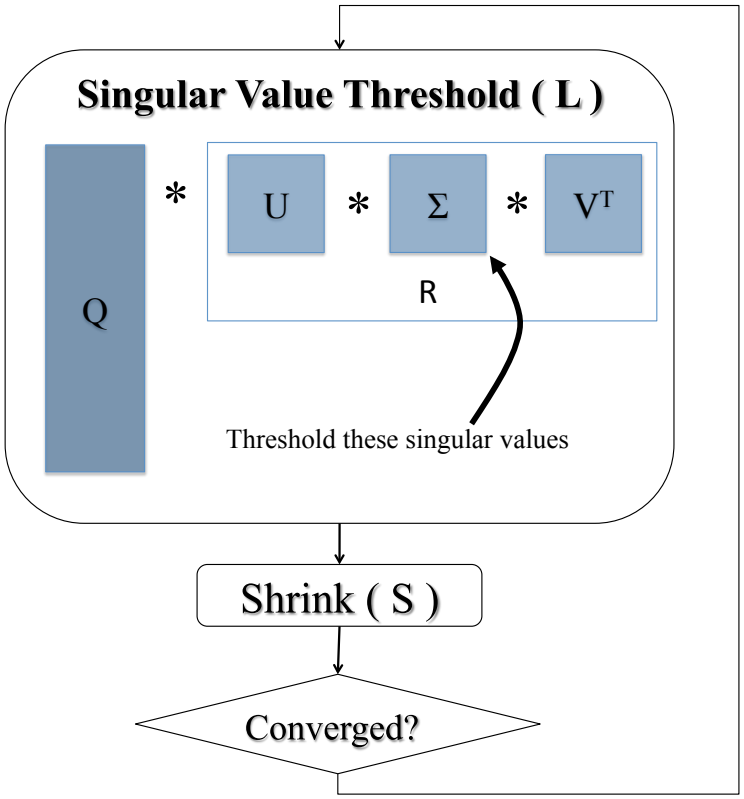
Idea: use Robust PCA algorithm [CLMW09] to subtract constant background from the action of a surveillance video

Given a matrix M whose columns represent frames, compute

$$M = L + S$$

where L is low-rank and S is sparse

Example Application: Video Background Subtraction



Compute:

$$M = L + S$$

where L is low-rank and S is sparse

The algorithm works iteratively, each iteration requires a singular value decomposition (SVD)

- M is $110{,}000 \times 100$

Communication-avoiding algorithm provided $3\times$ speedup over best GPU implementation [ABDK11]

Overview of Divide & Conquer Algorithm for Nonsymmetric Eigenproblem

One step of divide and conquer:

1. Compute $\left(I + (A^{-1})^{2^k}\right)^{-1}$ implicitly
   - maps eigenvalues of A to 0 and 1 (roughly)
2. Compute rank-revealing decomposition to find invariant subspace
3. Output block-triangular matrix

$$A_{\text{new}} = U^* A U = \begin{bmatrix} A_{11} & A_{12} \\ \varepsilon & A_{22} \end{bmatrix}$$

- block sizes chosen to minimize norm of ε
- eigenvalues of A_{11} all lie outside unit circle, eigenvalues of A_{22} lie inside unit circle, subproblems solved recursively
- stable, but progress guaranteed only with high probability

Reduction Example: LU

It's easy to reduce matrix multiplication to LU:

$$T \equiv \begin{bmatrix} I & 0 & -B \\ A & I & 0 \\ 0 & 0 & I \end{bmatrix} = \begin{bmatrix} I & & \\ A & I & \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} I & 0 & -B \\ & I & A \cdot B \\ & & I \end{bmatrix} \equiv L \cdot U$$

- LU factorization can be used to perform matrix multiplication
- Communication lower bound for matrix multiplication applies to LU

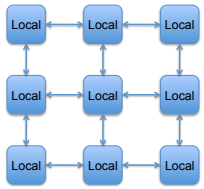
Reduction to Cholesky is a little trickier, but same idea [BDHS10]

Algorithms - Parallel $O(n^3)$ Linear Algebra

Algorithm	Reference	Factor exceeding lower bound for # words	Factor exceeding lower bound for # messages
Matrix Multiply	[Can69]	1	1
Cholesky	ScaLAPACK	$\log P$	$\log P$
Symmetric Indefinite	[BDD+12a] ScaLAPACK	proposed work $\log P$	proposed work $(N/P^{1/2}) \log P$
LU	[GDX11] ScaLAPACK	$\log P$ $\log P$	$\log P$ $(N/P^{1/2}) \log P$
QR	[DGHL12] ScaLAPACK	$\log P$ $\log P$	$\log^3 P$ $(N/P^{1/2}) \log P$
SymEig, SVD	[BDK12a] ScaLAPACK	proposed work $\log P$	proposed work $N/P^{1/2}$
NonsymEig	[BDD12b] ScaLAPACK	$\log P$ $P^{1/2} \log P$	$\log^3 P$ $N \log P$

*This table assumes that *one* copy of the data is distributed evenly across processors

Red = not optimal



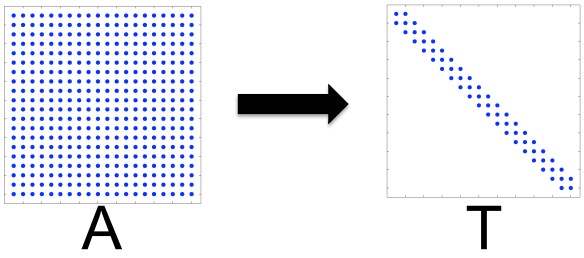
Back

Symmetric Eigenproblem and SVD via SBR

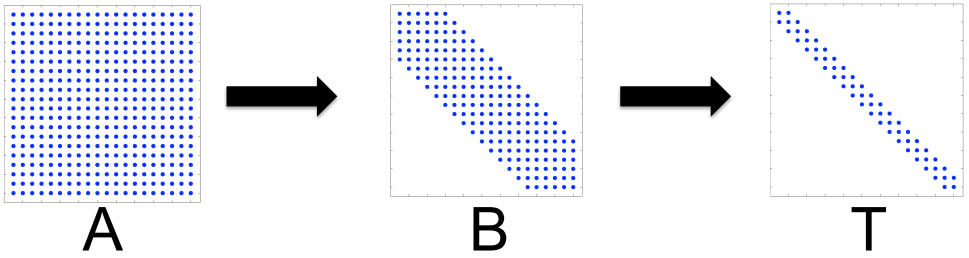
We're solving the symmetric eigenproblem via reduction to tridiagonal form

- Conventional approach (e.g. LAPACK) is direct tridiagonalization
- Two-phase approach reduces first to band, then band to tridiagonal

Direct:



Two-step:



- first phase can be done efficiently
- second phase is trickier, requires successive band reduction (SBR) [BLS00]
  - involves "bulge-chasing"
  - we've improved it to reduce communication [BDK12b]

Back

Communication-Avoiding SBR - theory

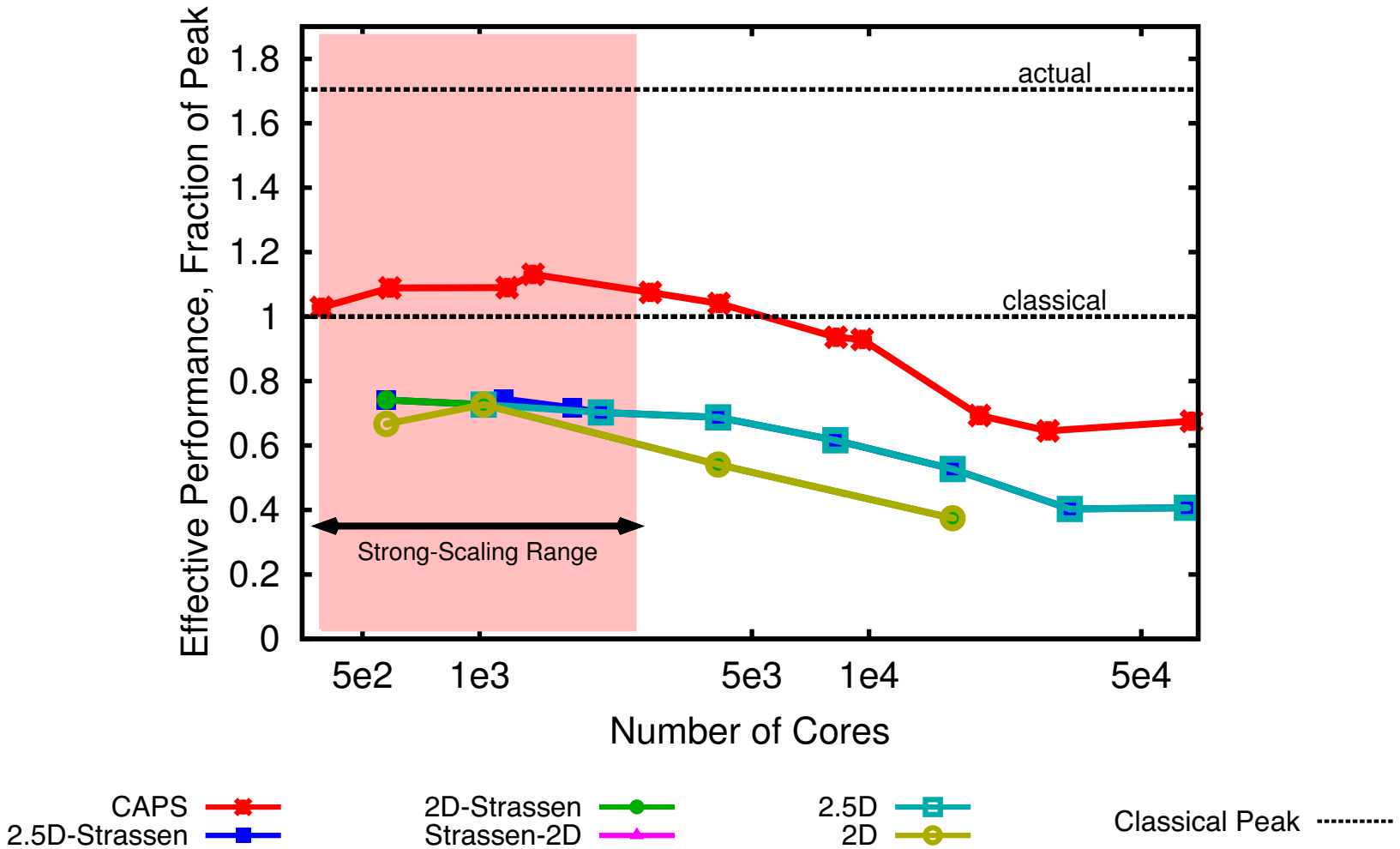
	Flops	Words Moved	Data Re-use
Schwarz	$4n^2b$	$O(n^2b)$	$O(1)$
M-H	$6n^2b$	$O(n^2b)$	$O(1)$
B-L-S*	$5n^2b$	$O(n^2 \log b)$	$O\left(\frac{b}{\log b}\right)$
CA-SBR†	$5n^2b$	$O\left(\frac{n^2b^2}{M}\right)$	$O\left(\frac{M}{b}\right)$

*with optimal parameter choices
†assuming $1 \leq b \leq \sqrt{M}/3$

Back

Performance of CAPS on large problems

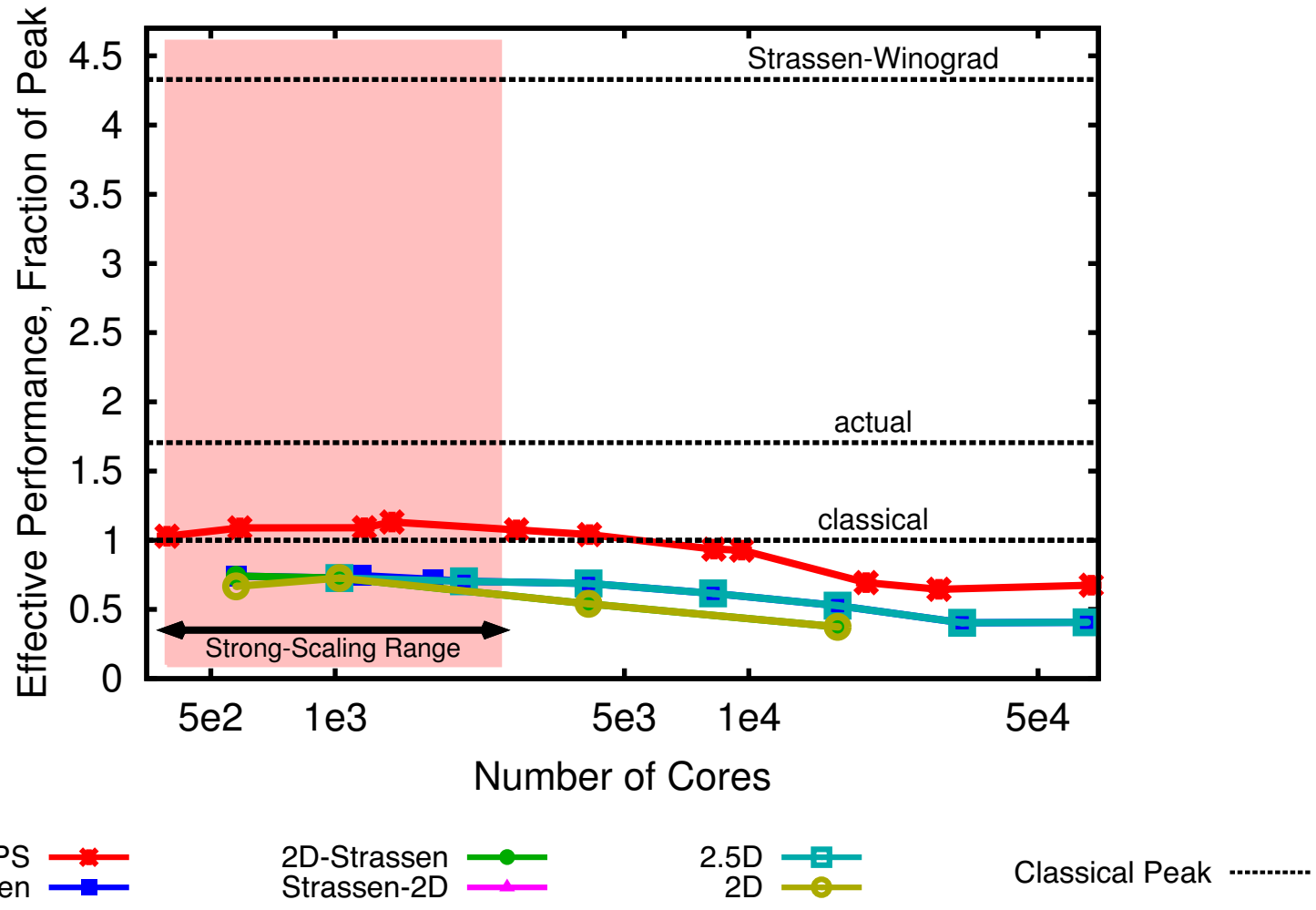
Strong-scaling on Intrepid (IBM BG/P), $n = 65{,}856$.



Back

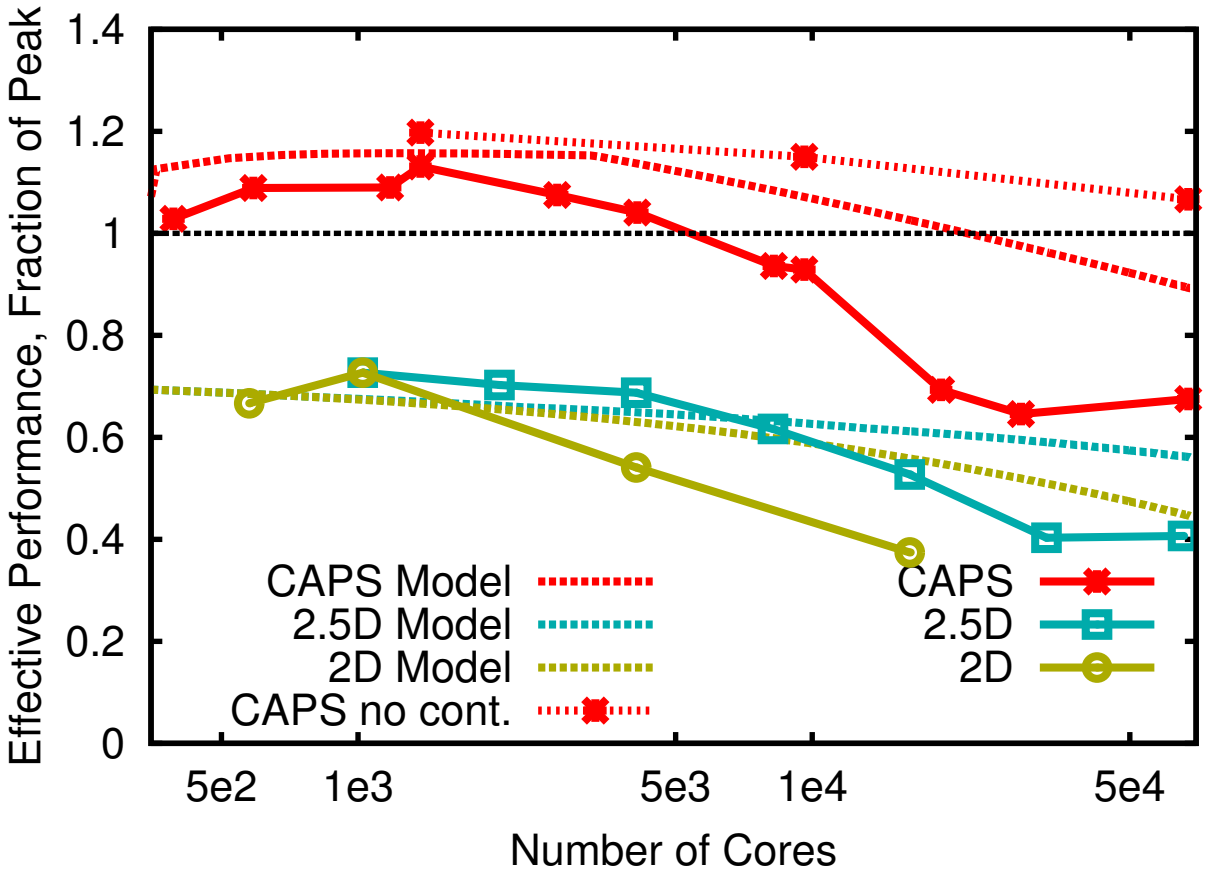
Performance of CAPS on large problems

Strong-scaling on Intrepid (IBM BG/P), $n = 65{,}856$.



Back

Performance: Model vs Actual



Comparison of the parallel models with the algorithms in strong scaling of matrix dimension $n = 65{,}856$ on Intrepid.

Back

References I

J. O. Aasen.

On the reduction of a symmetric matrix to tridiagonal form.
BIT Numerical Mathematics, 11:233–242, 1971.
10.1007/BF01931804.

M. Anderson, G. Ballard, J. Demmel, and K. Keutzer.

Communication-avoiding QR decomposition for GPUs.
In *Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium*, IPDPS '11, pages 48–58, Washington, DC, USA, 2011. IEEE Computer Society.

G. Ballard, D. Becker, J. Demmel, J. Dongarra, A. Druinsky, I. Peled, O. Schwartz, S. Toledo, and I. Yamazaki.

Implementing a blocked AasenÕs algorithm with a dynamic scheduler on multicore architectures, 2013.
To appear.

G. Ballard, J. Demmel, A. Druinsky, I. Peled, O. Schwartz, and S. Toledo.

Communication avoiding symmetric indefinite factorization, 2012.
In preparation.

G. Ballard, J. Demmel, and I. Dumitriu.

Communication-optimal parallel and sequential nonsymmetric eigenvalue algorithm, 2012.
In preparation.

G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz.

Brief announcement: strong scaling of matrix multiplication algorithms and memory-independent communication lower bounds.
In *Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 77–79, New York, NY, USA, 2012. ACM.

References II

G. Ballard, J. Demmel, O. Holtz, B. Lipshitz, and O. Schwartz.

Communication-optimal parallel algorithm for Strassen's matrix multiplication.
In *Proceedings of the 24th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 193–204, New York, NY, USA, 2012. ACM.

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.

Communication-optimal parallel and sequential Cholesky decomposition.
SIAM Journal on Scientific Computing, 32(6):3495–3523, 2010.

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.

Graph expansion and communication costs of fast matrix multiplication: regular submission.
In *Proceedings of the 23rd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '11, pages 1–12. ACM, 2011.

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.

Graph expansion and communication costs of fast matrix multiplication.
J. ACM, 59(6):32:1–32:23, December 2012.

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.

Sequential communication bounds for fast linear algebra.
Technical Report EECS-2012-36, UC Berkeley, March 2012.

G. Ballard, J. Demmel, and N. Knight.

Avoiding communication in the symmetric eigenproblem and SVD, 2012.
In preparation.

References III

G. Ballard, J. Demmel, and N. Knight.

Communication avoiding successive band reduction.
In *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming*, PPoPP '12, pages 35–44, New York, NY, USA, 2012. ACM.

C. Bischof, B. Lang, and X. Sun.

A framework for symmetric band reduction.
ACM Trans. Math. Soft., 26(4):581–601, December 2000.

L. Cannon.

A cellular computer to implement the Kalman filter algorithm.
PhD thesis, Montana State University, Bozeman, MN, 1969.

Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright.

Robust principal component analysis?
arXiv preprint arXiv:0912.3599, 2009.

J. Demmel, L. Grigori, M. Hoemmen, and J. Langou.

Communication-optimal parallel and sequential QR and LU factorizations.
SIAM Journal on Scientific Computing, 34(1):A206–A239, 2012.

L. Grigori, J. Demmel, and H. Xiang.

CALU: A communication optimal LU factorization algorithm.
SIAM Journal on Matrix Analysis and Applications, 32(4):1317–1350, 2011.

E. Solomonik and J. Demmel.

Communication-optimal parallel 2.5D matrix multiplication and LU factorization algorithms.
In Emmanuel Jeannot, Raymond Namyst, and Jean Roman, editors, *Euro-Par 2011 Parallel Processing*, volume 6853 of *Lecture Notes in Computer Science*, pages 90–109. Springer Berlin / Heidelberg, 2011.

Avoiding Communication in Dense Linear Algebra

Grey Ballard

Thank You!

www.eecs.berkeley.edu/~ballard
http://bebop.cs.berkeley.edu